WEST Search History

DATE: Friday, September 05, 2003

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT; PLUR=YES; OP=ADJ			
L11	L10 and l1	6	L11
L10	l6 and L9	30	L10
L9	l5 and L8	57	L9
L8	internet or network	271601	L8
L7	l3 and L6	2	L7
L6	l4 and L5	31	L6
L5	evb or enterprise java bean$	152	L5
L4	client and server	18096	L4
L3	behavior$ same l1	114	L3
L2	process$4 and (business adj2 data)	1242	L2 /
L1	lifecycle or life cycle	8031	L1 ✓

END OF SEARCH HISTORY

WEST

☐ Generate Collection Print

L11: Entry 3 of 6 File: USPT Jan 7, 2003

DOCUMENT-IDENTIFIER: US 6505342 B1
TITLE: System and method for functional testing of distributed, component-based software

Brief Summary Text (8):
As a result, developers are implementing large numbers of components ranging from relatively simple graphical user interface (GUI) components to sophisticated server-side application logic.

Brief Summary Text (11):
Basically, as developers are delivering these complex, server-side components, they must also ensure that each component is delivered with a concise and unambiguous definition of its interfaces, and the legal order in which operations may be invoked on them. Component interfaces and their protocol specifications are being described or modeled in a variety of ways. For example, in the case of the Enterprise Java Beans Specification, this is achieved through contracts and UML Sequence Diagrams (also known as Message Sequence Charts). While a Sequence Diagram is useful at describing a specific interaction scenario, it may require a large number of such diagrams to completely specify the interaction of a complex component with its client(s).

Detailed Description Text (4):
Referring now to FIG. 1, a flow diagram depicts an example of the method of operation for the functional testing of a component-based application according to one aspect of the present invention. It should be understood that the elements shown in the FIGS. may be implemented in various forms of hardware, software or combinations thereof. Preferably, these elements are implemented in software on one or more appropriately programmed general purpose digital computers having a processor and memory and input/output interfaces. A technique is employed for the use of UML-based state machine representations based on visual modeling tools, such as the Unified Modeling Language (UML), in modeling the dynamic behavior of components as well as the communications between them. UML is a general-purpose visual modeling language that is used to specify, visualize, construct and document the artifacts of a software system. An individual UML-based state machine representation 100 (hereinafter, "state machine") can be used to describe the dynamic behavior of an individual component or object over time by modeling its lifecycle.

Detailed Description Text (8):
FIG. 3 illustrates an example of a UML-based state machine representation for the Transmitter component 204 shown in FIG. 2 above. A state machine diagram is a concise and compact way of describing the dynamic behavior of a component over time by modeling its lifecycle. The key elements described in a state machine are states, transitions, events, and actions.

Detailed Description Text (88):
With an attribute definition block, any attributes can be defined in the target programming language, to be used within a test case. An example: Testcase TC1{ Define{ Attribute{ short counter; SampleServer server; } } }

Detailed Description Text (89):
The above example defines two attributes in the syntax of the target language, the

attribute counter of type short and the attribute server of type SampleServer.

Detailed Description Text (91):
The initialization block will be executed before the test case body. It includes any necessary initialization code. A common piece of code is one that acquires a reference to the server-based component object to be tested. Another task might be to register a sink interface with the server. The initialization block may include a set of test statements. With the help of the action and observe statements, initialization code that exemplifies asynchronous behavior can be placed in the initialization block.

Detailed Description Text (94):
The cleanup block will be executed after the test case body is completed. Any cleanup work can be done in this block; for example, deregistering of a sink interface from a server component.

Detailed Description Text (176):
As events are usually used for one way communication from server to the client and therefore usually don't have any out-parameters or return types, ITL doesn't provide a means to set values for out-parameters of sink methods or return values. Still this could be achieved by subclassing the sink objects generated out of the IDL file with specific behavior.

Detailed Description Text (282):
It is to be appreciated by those of ordinary skill in the art that the present invention may preferably be used for testing components that use middleware, such as CORBA and COM/DCOM. For example, an E-commerce application (used for shopping on the Internet) is made up of three tiers: a browser-based interface where a user types in an item search request, which is sent as an HTTP (Hyper Text Transfer Protocol) to a web server computer (tier 1), which then passes it on to a middle, or business logic tier (tier 2), which is where the application logic reside and the components that we want to test are used. These components may be distributed (residing on different machines in that tier). Once they process the item search request, it is sent to the database (tier 3) where a final result is computed and sent back to the first tier and is viewed by the user. In this example, the present invention is directed towards testing the components executing on the middle tier (tier 2). Other applications are also contemplated.

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 6 of 6 returned.

☐ 1. Document ID: US 6591272 B1

L11: Entry 1 of 6 File: USPT Jul 8, 2003

US-PAT-NO: 6591272
DOCUMENT-IDENTIFIER: US 6591272 B1

TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

☐ 2. Document ID: US 6557009 B1

L11: Entry 2 of 6 File: USPT Apr 29, 2003

US-PAT-NO: 6557009
DOCUMENT-IDENTIFIER: US 6557009 B1

TITLE: Environmental permit web portal with data validation capabilities

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

---

☐ 3. Document ID: US 6505342 B1

L11: Entry 3 of 6 File: USPT Jan 7, 2003

US-PAT-NO: 6505342
DOCUMENT-IDENTIFIER: US 6505342 B1

TITLE: System and method for functional testing of distributed, component-based software

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw. Desc | Image

---

☐ 4. Document ID: US 6292933 B1

L11: Entry 4 of 6 File: USPT Sep 18, 2001

US-PAT-NO: 6292933
DOCUMENT-IDENTIFIER: US 6292933 B1

**** See image for Certificate of Correction ****

TITLE: Method and apparatus in a data processing system for systematically serializing complex data structures

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

☐ 5. Document ID: US 6199195 B1

L11: Entry 5 of 6 File: USPT Mar 6, 2001

US-PAT-NO: 6199195
DOCUMENT-IDENTIFIER: US 6199195 B1

TITLE: Automatically generated objects within extensible object frameworks and links to enterprise resources

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

☐ 6. Document ID: US 6167564 A

L11: Entry 6 of 6 File: USPT Dec 26, 2000

US-PAT-NO: 6167564
DOCUMENT-IDENTIFIER: US 6167564 A

TITLE: Software system development framework

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

Generate Collection | Print

Term	Documents
(10 AND 1).USPT.	6
(L10 AND L1).USPT.	6

Display Format: TI | Change Format

Previous Page   Next Page